1. A cache memory, comprising:

a last-in-first-out (LIFO) memory, for caching lines of data implicated by destination addresses of push instructions, said LIFO memory having a top entry for storing a cache line implicated by a destination address of a most recent push instruction;

an input, coupled to said LIFO memory, for indicating that an instruction requesting data from the cache memory is a pop instruction specifying a source address of said data; and

logic, coupled to said input, for controlling said LIFO memory to speculatively provide said data to said pop instruction from said top entry in response to said input, prior to determining whether said source address matches an address of said cache line stored in said top entry.

2. The cache memory of claim 1, further comprising:

a comparator, coupled to said logic, for comparing said source address with said address of said cache line stored in said top entry, after said logic controls said LIFO memory to speculatively provide said data to said pop instruction.

3. The cache memory of claim 2, further comprising:

an output, coupled to said logic, for indicating that said LIFO memory incorrectly speculatively provided said data to said pop instruction, if said comparator indicates said source address of said pop instruction does not match said address of said cache line stored in said top entry.

4. The cache memory of claim 3, wherein said output indicates an exception condition.

5. The cache memory of claim 1, wherein said pop instruction comprises an x86 POP instruction.

6. The cache memory of claim 1, wherein said pop instruction comprises an x86 RET instruction.

7. The cache memory of claim 1, wherein said pop instruction comprises an x86 LEAVE instruction.

8. The cache memory of claim 1, wherein said pop instruction comprises an instruction for popping said data from a stack in a system memory coupled to a microprocessor comprising the cache memory.

9. The cache memory of claim 8, wherein said data popped from said stack memory comprises a return address.

10. The cache memory of claim 8, wherein said cache line comprises a number of bytes of data equal to a smallest number of bytes transferred between the cache memory and said system memory.

11. The cache memory of claim 8, wherein said destination address of said most recent push instruction comprises an address in said stack in said system memory.

12. The cache memory of claim 8, wherein said source address of said pop instruction comprises an address in said stack in said system memory.

13. The cache memory of claim 1, wherein said cache line comprises a first number of bytes of data, wherein said data provided to said pop instruction comprises a second number of bytes, wherein said first number of bytes is at least twice as much as said second number of bytes.

14. The cache memory of claim 1, wherein the cache memory maintains coherency of said lines of data cached therein with respect to other memories in a system comprising the cache memory.

15. The cache memory of claim 1, further comprising:

an offset, coupled for provision to said logic, for specifying a location within said top entry cache line of data specified by said most recent push instruction.

16. The cache memory of claim 15, wherein said logic controls said LIFO memory to speculatively provide said data from said location within said top entry cache line of data specified by said offset.

17. The cache memory of claim 16, wherein if updating said offset in response to said pop instruction causes said offset to wrap beyond said top entry cache line, said logic causes said LIFO memory to shift said top entry out of said LIFO memory.

18. The cache memory of claim 15, wherein if updating said offset prior to storing said data specified by said most recent push instruction causes said offset to wrap below said top entry cache line, said logic causes said LIFO memory to shift down, thereby creating a new top entry, wherein said logic stores said data specified by said most recent push instruction into said new top entry at said location specified by said offset.

19. The cache memory of claim 15, further comprising:

an arithmetic unit, coupled to said logic, for updating said offset.

20. The cache memory of claim 19, wherein said arithmetic unit updating said offset comprises said arithmetic unit adding to said offset an operand specified by an instruction, wherein said instruction specifies adding said operand to a stack pointer register of a microprocessor comprising the cache memory.

21. The cache memory of claim 20, wherein if updating said offset causes said offset to wrap beyond said top entry cache line, said logic causes said LIFO memory to shift said top entry out of said LIFO memory.

22. The cache memory of claim 1, wherein said most recent push instruction is most recent with respect to said cache lines of data stored in said LIFO memory.

23. The cache memory of claim 1, wherein said most recent push instruction is most recent with respect to not yet popped data stored in said LIFO memory.

24. The cache memory of claim 1, wherein said cache line implicated by said destination address of said most recent push instruction comprises data of a most recent push instruction whose data has not yet been popped from a stack in a system memory.

25. A cache memory, comprising:

an input, for indicating a type of an instruction reading data from the cache memory, said instruction specifying a source address of said data;

a last-in-first-out (LIFO) memory, coupled to said input, having a plurality of entries, each for storing a cache line of data implicated by a push instruction, having a top one of said plurality of entries for storing a cache line of data implicated by a most recent push instruction, wherein in response to said input indicating said instruction is a pop instruction, the cache memory provides data for said pop instruction from said top entry of said LIFO memory; and

a comparator, coupled to receive said source address, for generating an indication of whether said source address matches an address of said cache line stored in said top entry of said LIFO memory for determining whether said data provided for said pop instruction is correct data.

26. The cache memory of claim 25, wherein the cache memory provides said source data for said pop instruction from said top entry of said LIFO memory during a first clock cycle, wherein said comparator generates said indication during a second clock cycle.

27. The cache memory of claim 26, wherein said second clock cycle is subsequent to said first clock cycle.

28. The cache memory of claim 26, wherein said second clock cycle is not sooner than said first clock cycle.

29. The cache memory of claim 25, wherein said most recent push instruction is most recent with respect to said cache lines of data stored in said LIFO memory.

30. The cache memory of claim 25, further comprising:

an output, coupled to said comparator, for indicating said data provided for said pop instruction is incorrect data, if said source address does not match said address of said cache line stored in said top entry of said LIFO memory.

31. The cache memory of claim 25, further comprising:

an offset, coupled to said LIFO memory, for specifying a location within said cache line stored in said top entry of said data provided by the cache memory for said pop instruction.

32. The cache memory of claim 31, further comprising:

a multiplexer, coupled to said offset, for selecting said data from within said cache line based on said offset.

33. The cache memory of claim 25, wherein said source address and said address of said cache line stored in said top entry comprises physical memory addresses.

34. The cache memory of claim 25, wherein said LIFO memory also stores addresses of said lines of data cached therein, wherein the cache memory further comprises:

a plurality of comparators, coupled to receive said source address, for comparing said source address with said addresses of said lines of data cached in said LIFO memory, wherein said source address specifies a load address of load data specified by a load instruction.

35. The cache memory of claim 34, further comprising:

an output, coupled to said plurality of comparators, for indicating said load address hits in the cache memory if said plurality of comparators indicates said load instruction source address matches one of said addresses of said lines of data cached in said LIFO memory.

36. The cache memory of claim 25, a computer program product comprising a computer usable medium having computer readable program code causes the cache memory, wherein said computer program product is for use with a computing device.

37. The cache memory of claim 25, wherein a computer data signal embodied in a transmission medium comprising computer-readable program code provides the cache memory.

38. A cache memory, comprising:

a first plurality of storage elements, configured as a random access cache, for caching data specified by non-push instructions;

a second plurality of storage elements, configured as a stack cache, for caching lines of data implicated by destination addresses of push instructions, said stack cache having a top entry for storing a cache line implicated by a destination address of a most recent push instruction;

an input, for indicating that an instruction requesting data from the cache memory is a pop instruction specifying a source address of said data; and

logic, coupled to said input and to said first and second plurality of storage elements, for controlling said stack cache to speculatively provide said data to said pop instruction from said top entry in response to said input, prior to determining whether said source address matches an address of said cache line stored in said top entry.

39. A method for caching stack memory data, comprising:

storing into a cache memory, data specified by a push instruction and a destination address of the data in a stack memory;

decoding a pop instruction, subsequent to said storing;

generating a source address of data for the pop instruction, in response to said decoding;

comparing the source address with the destination address stored in the cache memory, after said generating; and

providing the data stored in the cache memory, without regard to said comparing, in response to said decoding.

40. The method of claim 39, further comprising:

determining the source address is not present in the cache memory, in response to said comparing; and

correcting for said providing the data, in response to said determining.

41. The method of claim 40, wherein said correcting comprises:

fetching from a stack memory other than the cache memory, correct data specified by the source address.

42. The method of claim 41, wherein said correcting further comprises:

generating a microprocessor exception, prior to said fetching.

43. The method of claim 39, wherein the cache memory is arranged as a last-in-first-out memory having a last-in entry, wherein said providing the data comprises:

providing the data stored in the last-in entry of the cache memory.

44. The method of claim 43, wherein the last-in entry is configured to store a cache line comprising a plurality of data words.

45. The method of claim 44, further comprising:

popping the last-in-first-out memory after said providing the data, if said speculatively providing comprises providing a last of the plurality of data words stored in the cache line stored in the last-in entry.

46. The method of claim 44, further comprising:

pushing down the last-in-first-out memory, prior to said storing, if the destination address misses in the cache memory.

47. The method of claim 39, wherein said providing is performed at least as soon as said comparing.

48. A method for speculatively providing data from a cache memory to a pop instruction, comprising:

storing first data specified by a destination address of a push instruction into a top entry of a last-in-first-out (LIFO) memory;

receiving a request from a pop instruction for second data specified by a source address, after said storing the first data;

providing the first data from the top entry of the LIFO memory without reference to the source address, in response to said receiving the request;

comparing the source address and the destination address; and

determining whether the first data is the second data, subsequent to said providing the first data, in response to said comparing.

49. The method of claim 48, further comprising:

generating an output for indicating the first data is not the second data, if the source address does not match the destination address.

50. The method of claim 49, further comprising:

providing the second data to the pop instruction after said generating the output.

51. The method of claim 50, further comprising:

executing an exception routine to perform said providing the second data.

52. The method of claim 48, further comprising:

performing a virtual to physical page address translation to generate the source address, no sooner than said providing the first data.

53. A computer data signal embodied in a transmission medium, comprising:

computer-readable program code for providing a cache memory, said program code comprising:

first program code for providing an input, for indicating a type of an instruction reading data from the cache memory, said instruction specifying a source address of said data;

second program code for providing a last-in-first-out (LIFO) memory, coupled to said input, having a plurality of entries, each for storing a cache line of data implicated by a push instruction, having a top one of said plurality of entries for storing a cache line of data implicated by a most recent push instruction, wherein in response to said input indicating said instruction is a pop instruction, the cache memory provides data for said pop instruction from said top entry of said LIFO memory; and

third program code for providing a comparator, coupled to receive said source address, for generating an indication of whether said source address matches an address of said cache line stored in said top entry of said LIFO memory for determining whether said data provided for said pop instruction is correct data.